Ingress Gateway without TLS Termination

4 minute read ✔ page test

The Securing Gateways with HTTPS task describes how to configure HTTPS ingress access to an HTTP service. This example describes how to configure HTTPS ingress access to an HTTPS service, i.e., configure an

ingress gateway to perform SNI passthrough, instead of TLS termination on incoming requests.

The example HTTPS service used for this task is a simple NGINX server. In the following steps you first deploy the NGINX service in your Kubernetes cluster. Then you configure a gateway to provide ingress access to the service via host `nginx.example.com`.

Generate client and server certificates and keys

For this task you can use your favorite tool to generate certificates and keys. The commands below use `openssl`

1. Create a root certificate and private key to sign the certificate for your services:

   ```
   $ openssl req -x509 -sha256 -nodes -days 365 -newkey rsa:20
   48 -subj '/O=example Inc./CN=example.com' -keyout example.c
   om.key -out example.com.crt
   ```

2. Create a certificate and a private key for `nginx.example.com`:

```
$ openssl req -out nginx.example.com.csr -newkey rsa:2048 -
nodes -keyout nginx.example.com.key -subj "/CN=nginx.exampl
e.com/O=some organization"
$ openssl x509 -req -sha256 -days 365 -CA example.com.crt -
CAkey example.com.key -set_serial 0 -in nginx.example.com.c
sr -out nginx.example.com.crt
```

Deploy an NGINX server

1. Create a Kubernetes Secret to hold the server's certificate.

```
$ kubectl create secret tls nginx-server-certs --key nginx.
example.com.key --cert nginx.example.com.crt
```

2. Create a configuration file for the NGINX server:

```
$ cat <<\EOF > ./nginx.conf
events {
}

http {
  log_format main '$remote_addr - $remote_user [$time_local
]  $status '
  '"$request" $body_bytes_sent "$http_referer" '
  '"$http_user_agent" "$http_x_forwarded_for"';
  access_log /var/log/nginx/access.log main;
  error_log  /var/log/nginx/error.log;

  server {
    listen 443 ssl;
```

```
    root /usr/share/nginx/html;
    index index.html;

    server_name nginx.example.com;
    ssl_certificate /etc/nginx-server-certs/tls.crt;
    ssl_certificate_key /etc/nginx-server-certs/tls.key;
  }
}
EOF
```

3. Create a Kubernetes ConfigMap to hold the configuration of the NGINX server:

```
$ kubectl create configmap nginx-configmap --from-file=ngin
x.conf=./nginx.conf
```

4. Deploy the NGINX server:

```
$ cat <<EOF | istioctl kube-inject -f - | kubectl apply -f
-
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 443
    protocol: TCP
  selector:
    run: my-nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
```

```
selector:
  matchLabels:
    run: my-nginx
replicas: 1
template:
  metadata:
    labels:
      run: my-nginx
  spec:
    containers:
    - name: my-nginx
      image: nginx
      ports:
      - containerPort: 443
      volumeMounts:
      - name: nginx-config
        mountPath: /etc/nginx
        readOnly: true
      - name: nginx-server-certs
        mountPath: /etc/nginx-server-certs
```

```
          readOnly: true
      volumes:
      - name: nginx-config
        configMap:
          name: nginx-configmap
      - name: nginx-server-certs
        secret:
          secretName: nginx-server-certs
EOF
```

5. To test that the NGINX server was deployed successfully, send a request to the server from its sidecar proxy without checking the server's certificate (use the `-k` option of `curl`). Ensure that the server's certificate is printed correctly, i.e., `common name (CN)` is equal to `nginx.example.com`.

```
$ kubectl exec "$(kubectl get pod  -l run=my-nginx -o jsonp
ath={.items..metadata.name})" -c istio-proxy -- curl -sS -v
 -k --resolve nginx.example.com:443:127.0.0.1 https://nginx
.example.com
...
SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
ALPN, server accepted to use http/1.1
Server certificate:
  subject: CN=nginx.example.com; O=some organization
  start date: May 27 14:18:47 2020 GMT
  expire date: May 27 14:18:47 2021 GMT
  issuer: O=example Inc.; CN=example.com
  SSL certificate verify result: unable to get local issuer
 certificate (20), continuing anyway.

> GET / HTTP/1.1
> User-Agent: curl/7.58.0
> Host: nginx.example.com
...
< HTTP/1.1 200 OK
```

```
< Server: nginx/1.17.10
...
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

Configure an ingress gateway

1. Define a `Gateway` with a `server` section for port 443.

Note the PASSTHROUGH TLS mode which instructs the gateway to pass the ingress traffic AS IS, without terminating TLS.

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: mygateway
spec:
  selector:
    istio: ingressgateway # use istio default ingress gatew
ay
  servers:
  - port:
      number: 443
      name: https
      protocol: HTTPS
    tls:
      mode: PASSTHROUGH
    hosts:
    - nginx.example.com
EOF
```

2. Configure routes for traffic entering via the `Gateway`:

```
$ kubectl apply -f - <<EOF
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: nginx
spec:
  hosts:
  - nginx.example.com
  gateways:
  - mygateway
  tls:
  - match:
    - port: 443
      sniHosts:
      - nginx.example.com
    route:
```

```
    - destination:
        host: my-nginx
        port:
          number: 443
EOF
```

3. Follow the instructions in Determining the ingress IP and ports to define the `SECURE_INGRESS_PORT` and `INGRESS_HOST` environment variables.
4. Access the NGINX service from outside the cluster. Note that the correct certificate is returned by the server and it is successfully verified (*SSL certificate verify ok* is printed).

```
$ curl -v --resolve "nginx.example.com:$SECURE_INGRESS_PORT
:$INGRESS_HOST" --cacert example.com.crt "https://nginx.exa
mple.com:$SECURE_INGRESS_PORT"
Server certificate:
  subject: CN=nginx.example.com; O=some organization
  start date: Wed, 15 Aug 2018 07:29:07 GMT
  expire date: Sun, 25 Aug 2019 07:29:07 GMT
  issuer: O=example Inc.; CN=example.com
  SSL certificate verify ok.

  < HTTP/1.1 200 OK
  < Server: nginx/1.15.2
  ...
  <html>
  <head>
  <title>Welcome to nginx!</title>
```

Cleanup

1. Remove created Kubernetes resources:

```
$ kubectl delete secret nginx-server-certs
$ kubectl delete configmap nginx-configmap
$ kubectl delete service my-nginx
$ kubectl delete deployment my-nginx
$ kubectl delete gateway mygateway
$ kubectl delete virtualservice nginx
```

2. Delete the certificates and keys:

```
$ rm example.com.crt example.com.key nginx.example.com.crt
nginx.example.com.key nginx.example.com.csr
```

3. Delete the generated configuration files used in this example:

```
$ rm ./nginx.conf
```